Grado en Ingeniería Informática

Trabajo de Final de Grado

Aplicación web con servicio telefónico VoIP para la atención a clientes de una empresa

Autor:
María Mañes Velasco

Supervisor:
Aitor Guerrero Gómez
Tutor académico:
María José Aramburu Cabo

Fecha de lectura: 16 de Septiembre de 2020
Curso académico 2019/2020

Resumen

En este documento se detalla la especificación, planificación, diseño e implementación del proyecto de fin de grado realizado por María Mañes Velasco, que hizo la estancia en prácticas en la empresa Nayar Systems de Castellón durante 300 horas.

El proyecto realizado es una aplicación *frontend* que consiste en una página web dentro de la aplicación de soporte de la empresa y desde la cual se puede realizar llamadas, almacenar la información recogida en llamadas y visualizar información relevante del cliente durante llamadas VoIP (Voice over Internet Protocol). La aplicación se conectará mediante el *backend* a la base de datos y al ERP de la empresa para buscar o almacenar información recogida.

Para la realización de este proyecto se ha seguido la metodología Kanban, con la cual, se ha conseguido alcanzar los resultados satisfactoriamente en el tiempo planificado.

Palabras clave

VoIP (Voice over Internet Protocol), ERP (Enterprise Resource Planning), SIP (Session Initiation Protocol)

Keywords

VoIP (Voice over Internet Protocol), ERP (Enterprise Resource Planning), SIP (Session Initiation Protocol)

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Índice del capítulo

1.1. Contexto y motivación del proyecto

Este proyecto ha sido realizado por la estudiante María Mañes Velasco en la empresa Nayar Systems, ubicada en Castellón de la Plana, como trabajo de fin de grado, parte de la asignatura "EI1054 - Prácticas externas y trabajo de fin de grado" del grado universitario en Ingeniería Informática de la Universidad Jaume I de Castellón, con especialización en el itinerario de Sistemas de Información.

1.1.1. Empresa

Nayar Systems es una empresa especializada en la ingeniería de telecomunicaciones. Se basa en la tecnología de Internet de las cosas (IoT) [1]. La empresa cuenta con tres marcas comerciales, que son las siguientes.

- **Advertisim:** Centrada en el desarrollo de un dispositivo con tecnología 3G y wifi capaz de explotar contenidos publicitarios en tiempo real. Este dispositivo, además, incorpora una pantalla donde muestra información útil.
- **Net4Machines:** Se estructura una red privada y virtual (VPN) desarrollada para permitir la conexión entre todo tipo de dispositivos por medio de Internet. El objetivo de N4M

es dotar al usuario de un control total sobre los dispositivos con los que cuenta. Ofrece, en tiempo real, monitorización total de la red, de modo que permite al usuario acceder y almacenar la información acerca de los dispositivos conectados.

- **72horas:** Especializada en la comunicación M2M (*Machine to Machine*) [11], se centra en el cumplimiento de la normativa EN 81-28. Esta norma establece que los ascensores deben disponer de un sistema de alarma que permita una comunicación bidireccional entre los usuarios atrapados en un ascensor y el servicio de rescate. De esta forma, Nayar se encarga de proporcionar una línea telefónica por medio de una tarjeta SIM, de la que el pasajero puede hacer uso en caso de emergencia y el técnico se informa del estado del ascensor cada 72 horas, tal y como esta normativa índica.

Además, Nayar Systems cuenta con un departamento denominado "Garaje". Este departamento está destinado a la creación, la innovación y al desarrollo tecnológico, y permite a los empleados adquirir conocimientos que enriquecen el carácter innovador de la empresa, es decir, es un departamento donde es posible llevar a cabo ideas innovadoras propuestas por los empleados.

Este proyecto se ha desarrollado para el departamento de soporte técnico de la empresa, que se comunica con los clientes para resolver dudas, solucionar problemas, informar o actualizar información de los dispositivos o la empresa. Además, a veces recibe las llamadas vacías de las telealarmas, que son llamadas que generan los dispositivos instalados en los ascensores.

1.1.2. Motivación del proyecto

Actualmente el departamento de soporte técnico debe utilizar diversas plataformas para atender una llamada y documentarla correctamente: una para realizar y recibir las llamadas, otra ventana con el ERP donde se registra la llamada y otra donde se documentan las posibles incidencias. Se quiere realizar una aplicación que cubra todas estas necesidades juntas. Por lo tanto, ha de ser posible llamar, documentar una llamada con la información relevante y graficar por características las llamadas recibidas por el cliente. Esto reducirá el tiempo invertido en documentar una llamada, siendo que se realizará todo en la misma página y solucionará problemas de hacer esperar al cliente si alguna página no carga. También permitirá una atención más personalizada y un mayor número de llamadas.

1.2. Objetivos del proyecto

El objetivo es crear una página web dentro de la aplicación web de la empresa que agilice el proceso de realizar o recibir llamadas, documentarlas y además mostrar información relevante sobre llamadas anteriores del cliente o la empresa con la que se habla.

El objetivo es centralizar las tareas que se realizan en la atención al cliente en una sola página. Por ello, en la página web se deben realizar llamadas a clientes, inicialmente marcando el número de teléfono, pero posteriormente se añadirá una búsqueda por nombre en la agenda

de teléfonos de la empresa. Se deben recibir llamadas donde aparece el nombre del cliente que llama y a la empresa a la que representa, para conseguir un acercamiento a la persona. Por ésta misma razón, se desea mostrar una gráfica de las últimas llamadas tanto del cliente como de la empresa, para poder actuar con antelación ante posibles problemas. Además de guardar siempre un registro de las llamadas, con la duración, la solución que se le haya adoptado para resolver la llamada, y entre otras, la posibilidad de añadir un resumen o comentario adicional que resulte de utilidad.

Para ello se desarrolla una página web que utiliza la librería SIP.js para realizar y recibir las llamadas mediante VoIP (Voice over Internet Protocol). Estás llamadas serán registradas en el ERP para poder consultar la información en cualquier momento, además mostrará una gráfica de las llamadas realizadas en los últimos 7 días por el cliente o la empresa, mostrando la categoría en la que se clasificaron.

1.3. Situación previa y desafíos

Como se ha mencionado anteriormente, hasta el momento la empresa no disponía de una herramienta que realizara todo conjuntamente, sino que contaba con diferentes aplicaciones que ejecutaban independientemente las tareas.

Por lo tanto, el desafío fue definir los requisitos y desarrollar el proyecto de forma que unifique en una sola página web todas las herramientas utilizadas actualmente, cubriendo las necesidades que ahora están diseminadas en diferentes aplicaciones, y sobre todo, que sea fácil de utilizar e intuitiva para los usuarios, dentro del plazo de tiempo de la estancia en prácticas. El mayor reto tecnológico ha sido integrar el servidor con la biblioteca SIP.js de forma que se conectara y que se escucharan las llamadas en ambas direcciones.

Uno de los mayores contratiempos fue el estado de alerta que se activó en el país durante la estancia en prácticas por la situación del COVID-19, que produjo un cambio en el modo de realizar las prácticas de presenciales a telemáticas, cambiando las reuniones y consultas presenciales por videoconferencias.

1.4. Estructura de la memoria

Para organizar mejor el contenido de este documento, comentaré el contenido de cada capítulo. En primer lugar, el capítulo 2 hace referencia a una descripción detallada de las herramientas y las tecnologías utilizadas para desarrollar el proyecto.

En segundo lugar, el capítulo 3 hace referencia a una descripción mucho más detallada del proyecto, respecto a su alcance, requisitos y la arquitectura tecnológica utilizada.

En tercer lugar, el capítulo 4 habla acerca de la información relacionada con la metodología utilizada para desarrollar el proyecto. Muestra la planificación, la estimación de recursos y cómo ha sido el seguimiento del proyecto.

A continuación, en el capítulo 5 se detalla el análisis y el diseño de la aplicación, hablando y mostrando además el diseño de la interfaz.

Después, el capítulo 6 muestra la implementación de dicha aplicación y las pruebas realizadas para comprobar su correcto funcionamiento.

Finalmente, el documento terminará con el capítulo 7, que trata los resultados del proyecto, conclusiones obtenidas tras el proceso de desarrollo, tanto técnicas como personales, y un apartado con posibles mejoras de la aplicación. Para terminar hay un apartado con la bibliografía que ha sido utilizada para llevar a cabo este proyecto.

Capítulo 2

Tecnologías y herramientas

Índice del capítulo

En este capítulo se explican las diferentes tecnologías y herramientas que se han utilizado para el desarrollo del proyecto y el motivo. Al ser el proyecto una funcionalidad añadida a la aplicación de la empresa que ya existía, esta nueva funcionalidad debe seguir el mismo criterio. Por lo tanto algunas de las tecnologías ya estaban definidas, como son, Vue.js, Go o Nexus. En cambio, la librería de SIP.js, fue elegida por la alumna, entre otras opciones que la empresa había propuesto tras documentarse.

2.1. Lenguajes

2.1.1. Go

Go, o Golang, es un lenguaje de programación concurrente y compilado, orientado a objetos, que pretende ser tan dinámico como Python pero con una sintaxis similar a C. Está desarrollado por Google y fue lanzado en 2009 [9].

Este lenguaje ha sido utilizado para la parte del *backend* del proyecto además de servir para hacer las consultas a Nexus, el software que se encarga de relacionar la aplicación con la base de datos haciendo consultas y enviando la información solicitada por el *frontend*. Más adelante se explica que esta herramienta se utiliza como biblioteca interna para conectarse con el ERP que hace de base de datos de las llamadas. El lenguaje Go fue elegido por la empresa por su simplicidad y la eficiencia que está demostrando. También decir que por ser tan nuevo, dado que está dando tan buenos resultados y por estar en continuo desarrollo con nuevas versiones, se espera que pueda dar una larga vida a los proyectos.

2.1.2. JavaScript

JavaScript es un lenguaje de programación orientado a objetos para realizar actividades complejas en una página web permitiendo mejoras en la interfaz del usuario [10]. Es un lenguaje dinámico, funcional y prototípico que surgió hace más de 20 años.

Este lenguaje ha sido utilizado en el proyecto para la parte de *frontend*, siendo utilizado por el *framework* de Vue.

2.2. Framework y librerías

2.2.1. Vue.js

Vue.js es un *framework* de JavaScript de código abierto del tipo Modelo-Vista-Modelo para construir interfaces de usuario, que está basado en AngularJS. En un mismo fichero encontramos HTML, JavaScript y el CSS de una manera separada y organizada. Vue utiliza una sintaxis de plantilla basada en HTML que permite vincular el DOM representado a los datos [7].

Este *framework* se ha utilizado para generar el *frontend* del proyecto, que se ha realizado con este *framework* porque las demás funcionalidades de la aplicación ya lo hacen. Se eligió por su sencillez y potencia para cargar únicamente los componentes necesarios. El *frontend* se encarga de enviar al *backend* las consultas pertinentes para mostrárselas al usuario.

2.2.2. Nexus

Nexus [18] es un servidor de mensajería de código abierto diseñado para permitir la comunicación y sincronización entre múltiples componentes distribuidos y está escrito en el lenguaje Go. Fue desarrollado por José Luis Aracil Gómez del Campo trabajador de la empresa Nayar Systems.

En el proyecto se utiliza como biblioteca interna para realizar el paso de tareas y mensajería, para conseguir una comunicación con Odoo, el ERP de la empresa, que se utiliza como base de datos de las llamadas y de los clientes. Las consultas son generadas desde el *backend*.

2.2.3. SIP.js

SIP.js es una biblioteca de JavaScript que ayuda a introducir el protocolo SIP en una aplicación WebRTC. SIP es un protocolo de señalización utilizado para iniciar, mantener y finalizar sesiones de comunicación en tiempo real que incluyen aplicaciones de voz, vídeo y mensajería. Se utiliza para controlar sesiones de comunicación multimedia en aplicaciones de telefonía por Internet, para llamadas de voz y vídeo, en sistemas telefónicos a través de redes de Protocolo de Internet (IP) [15].

Esta biblioteca fue elegida entre otras propuestas. Para esta decisión se tuvo en cuenta la información facilitada en las webs, contrastación de opiniones de otros usuarios y ver que cubría las necesidades requeridas en este caso y para una posible ampliación de funcionalidades. Se ha utilizado, para integrar SIP en la aplicación WebRTC, permitiendo realizar llamadas desde la web creada como nueva funcionalidad en la aplicación de la empresa. Esta se conecta con un servidor proporcionado y gestionado por la empresa, que también seleccionó como mejor opción para este proyecto.

2.3. Sistema de planificación de recursos empresariales (ERP)

2.3.1. Odoo

Odoo es un software de ERP integrado que cuenta con una versión de código abierto y otra versión comercial para las empresas. También cuenta con un sitio web donde gestionar toda la información de la empresa: facturación, contabilidad, fabricación y gestión de almacenes. Odoo utiliza la base de datos PostgreSQL. Este software se lanzó en 2004 y cuenta con 13 versiones hasta el momento [6].

Éste es el software que utiliza la empresa como ERP y actualmente como base de datos de las llamadas y los clientes. Se han tenido en cuenta otras bases de datos para almacenar las llamadas del nuevo proyecto, pero la decisión fue mantener Odoo como base de datos de las llamadas, así como para poder acceder a la información de los clientes y las llamadas realizadas anteriormente. Además de considerarse como una tecnología válida se evita una migración de los datos a una nueva plataforma.

2.4. Control de versiones

2.4.1. Git y Bitbucket

Git es un software de control de versiones. Su propósito es almacenar los archivos de los proyectos para que se registren los cambios de los archivos de computadora y en la nube, de forma que se puede coordinar el trabajo que varias personas realizan sobre archivos compartidos. De este modo, todos pueden acceder a ellos y modificar únicamente algunas líneas, de manera que varias personas pueden trabajar en un mismo fichero y después hacer una fusión de ambos, además de tener la posibilidad de cargar una versión anterior si fuera necesario [4].

Para conseguir una gestión más visual de las versiones, en Nayar Systems se utiliza, Bitbucket [3], donde se pueden ver las versiones subidas con información relevante como quién la subió, cuándo y un comentario resumiendo los cambios añadidos al código.

2.5. Entorno de desarrollo

2.5.1. Visual Studio Code

Visual Studio Code es un editor de código que además cuenta con un control de versiones Git integrado, un resaltado de sintaxis para ayudar a programar más visualmente y es compatible con varios lenguajes [17].

Este entorno es el recomendado por la empresa, pueden utilizarse otros, pero está considerado como uno de los mejores entornos de desarrollo. Es ampliamente utilizado por su sencillez, los numerosos *plugins* que facilitan el trabajo con ayudas como colores para cada tipo de objeto en el código, sugerencias en la tabulación del código para simplificar la visión de este, etc. En este proyecto se ha utilizado para programar tanto el *frontend* en JavaScript (Vue) como el *backend* en Go.

2.6. Gestión de proyecto

2.6.1. Jira

Jira es una herramienta en línea que sirve para administrar las tareas de un proyecto, el seguimiento de errores e incidencias y para la gestión operativa de proyectos. Forma parte de la empresa Atlassian y fue lanzado en 2002 [2].

Esta herramienta ha sido utilizada como ayuda para seguir la metodología de flujos de trabajo Kanban, como se explica en el apartado 4.1 de este documento. Esta metodología se caracteriza por estar enfocada a llevar a cabo determinadas tareas, una tras otra.

Capítulo 3

Descripción del proyecto

Índice del capítulo

El proyecto se ha desarrollado partiendo de una lista de requisitos iniciales facilitada por la empresa que indicaba los resultados que se debían obtener.

A lo largo de la estancia en prácticas, se realizaron dos reuniones con el que será "el cliente final" (el encargado de los empleados del departamento del área técnica de la empresa). En estas reuniones, estos requisitos fueron transformándose según las necesidades principales a cubrir y los problemas que pudieron surgir. Cabe añadir que el imprevisto estado de alerta en el país causó que se volvieran a revisar las tareas, puesto que el tiempo estimado para realizarlas aumentó.

3.1. Proceso de definición de requisitos

Al iniciar la estancia en prácticas el requisito era desarrollar un cliente SIP para un navegador web que se conectase a un servidor PBX (Private Branch Exchange) mediante el cual se pudiese realizar llamadas en ambos sentidos a través de la capa WebRTC, tanto realizarlas como recibirlas, y por supuesto que se escuchara en ambos sentidos. Este requisito fue alcanzado antes de hacer un mes, como más adelante se puede observar en el diagrama de Gannt final.

3.1.1. Requisitos finales del proyecto

Una vez desarrollado el cliente SIP y tras comprobar que las llamadas funcionaban correctamente, se realizaron dos reuniones con el "cliente final". La primera a principios de marzo donde se habló de posibles funcionalidades a añadir en la página web, y otra reunión tras el estado de alerta, a mediados de abril. En este momento ya se había conseguido realizar una de las funcionalidades habladas en la primera reunión, que consistía en que las llamadas fueran almacenadas en ERP de la empresa que funciona como base de datos, por lo cual se decidió realizar una funcionalidad más que fuera posible terminar durante el resto de horas que quedaban. Esta funcionalidad fue que se visualizara un gráfico con las llamadas de los últimos 7 días del cliente, al que finalmente también se le añadieron estos datos de la empresa, puesto que de una misma empresa no siempre llama la misma persona.

En resumen, la lista de requisitos funcionales del proyecto es la siguiente:

- Desarrollar un cliente SIP capaz de conectarse al servidor PBX
- Desarrollar la capa WebRTC
- Realizar llamadas
- Recibir llamadas
- Conseguir comunicación de audio y micro en ambos sentidos
- Almacenar registro de llamadas con información útil
- Representar gráficamente información de llamadas anteriores

A estos requisitos funcionales cabe añadir los requisitos tecnológicos explicados en el capitulo 2 donde se explican cuáles son utilizados y la razón por la que se eligieron para este proyecto.

3.2. Alcance

Con el alcance del proyecto indicamos qué sistemas externos serán afectados por el desarrollo, así como el departamento que utiliza el sistema. Se pueden diferenciar en diferentes tipos de alcance:

- Alcance funcional. Para poder acceder a la página se debe iniciar sesión en la web de soporte de la empresa. Se pueden realizar llamadas introduciendo el número de teléfono al que llamar, o aceptar o rechazar una llamada entrante. Durante la llamada aparece un botón que despliega una botonera de DTMF, un formulario donde se rellenan algunos datos automáticamente, como el número de teléfono, el nombre del cliente, el nombre de la empresa y la duración de la llamada, y otros datos que se introducen manualmente, como el tipo de la llamada, la categoría y el resultado. Además, existe la opción de escribir anotaciones. Esta información se envía al ERP para almacenarla tras terminar la llamada. Además, también aparecen dos gráficas con las llamadas de los últimos 7 días del cliente y de la empresa.

- Alcance organizativo. La aplicación está diseñada únicamente para ser utilizada por el departamento de soporte, que es el encargado de recibir y realizar llamadas, las gestiona y las registra.
- Alcance informático. La nueva página web se añade como funcionalidad a la aplicación ya existente de Nayar App para el departamento de soporte de la empresa, donde el *frontend* envía peticiones al *backend* y este las transforma en consultas que envía mediante Nexus a Odoo donde se almacena la información que devolverá para ser gestionada y mostrada al usuario.

3.3. Restricciones

El proyecto tiene algunas restricciones que se deben tener en cuenta, antes y durante su desarrollo, con el fin obtener el resultado esperado:

- La restricción temporal es que el proyecto debe estar finalizado en el periodo de prácticas acordado entre la empresa y la universidad, que son 300 horas de desarrollo en la oficina de la empresa asignada. Durante este tiempo hay un seguro que cubre al alumno por lo cual no puede ser extendido sin aviso.
- La restricción económica no existe ya que el estudiante desarrolla el proyecto de forma gratuita para la empresa y tanto el *hardware* como el *software* que se utiliza son propiedad de la empresa y las aplicaciones utilizadas son gratuitas.
- Las restricciones humanas, el proyecto se realiza únicamente por un desarrollador, el estudiante en prácticas, bajo la supervisión del encargado representando la empresa y una tutora asignada por la Universidad Jaume I para guiar a la alumna durante la estancia y el desarrollo de la memoria, respectivamente.
- Las restricciones tecnológicas y materiales, no se requiere nada con lo que la empresa no contase ya, como son las instalaciones, ordenador y aquel material de oficina que se pueda obviar, además de un teléfono móvil para realizar las pruebas oportunas de funcionamiento.

3.4. Arquitectura tecnológica

El proyecto se ha desarrollado partiendo del *framework* Vue.js, por lo que el patrón de arquitectura de software seguido ha sido el de Modelo Vista Modelo de Vista (MVVM) que se puede observar en la figura 3.1. Este patrón se caracteriza por tratar de desacoplar lo máximo posible la interfaz de usuario de la lógica de la aplicación [12].

Técnicamente, Vue.js se centra en el Modelo de Vista conectando así el Modelo con la Vista a través de enlaces bidireccionales. El objetivo es proporcionar con una API que sea lo más simple posible. Está diseñado para ser una capa de vista simple y flexible. En la figura 3.2 podemos observar qué parte representa cada una de las capas del patrón [13]. Este patrón

se caracteriza porque no requiere que el *backend* envíe nada para actualizar la vista. Ésta se actualiza automáticamente cuando tiene nueva información.

Figura 3.1: Patrón Modelo Vista Modelo de Vista.

La parte del Modelo de la Vista, que es el papel de Vue, es la que se encarga de comunicarse con el *backend* para enviar peticiones y gestionar la información que obtiene y que envía a la Vista donde se muestran al usuario. Esto sucede tanto cuando se realiza una llamada como cuando se recibe. Se busca el número de teléfono en el ERP con la intención de obtener información sobre el cliente, como su nombre, el de la empresa, y las llamadas realizadas anteriormente. Además, también se recibe una lista de características posibles para asignar a la llamada antes de guardarla. Se muestra esta información por pantalla y al registrar la llamada se envía la información introducida para registrarla en el ERP y poder acceder a ella posteriormente.

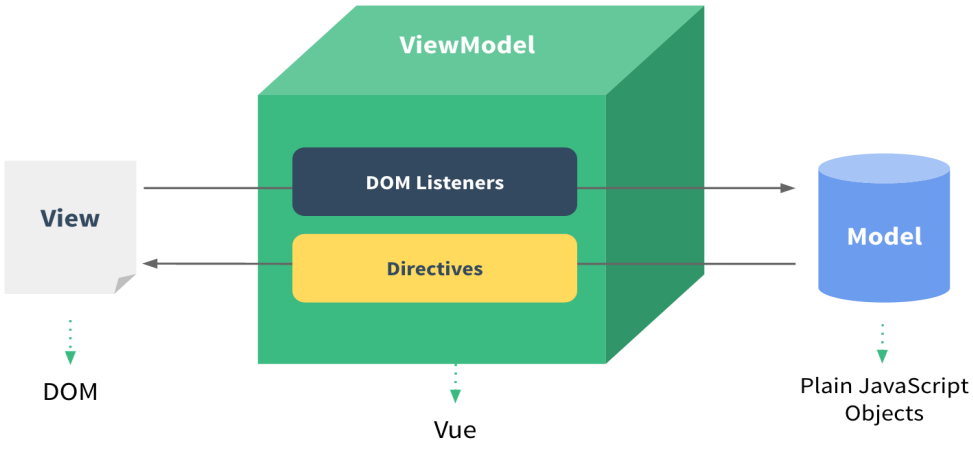

Figura 3.2: Conceptos de Vue según el patrón MVVM.

Otra parte importante para el proyecto, aunque no forma parte de sus objetivos, es el servidor de comunicaciones PBX (Private Branch Exchange), es decir, una central telefónica que se conecta directamente a la red pública de telefonía que gestiona además de las llamadas internas, las entrantes y salientes. Para este proyecto se ha utilizado el *software* llamado Issabel. Este servidor PBX es el encargado de conectar la interfaz web con el centro de llamadas, ver figura 3.3.

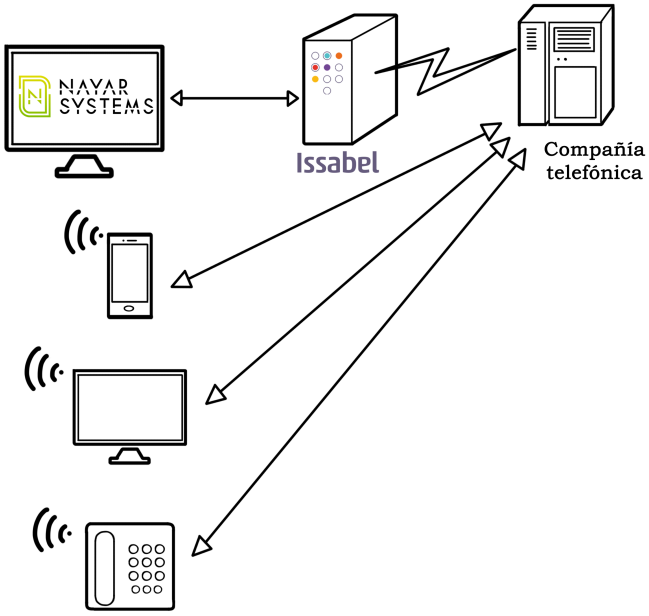

Figura 3.3: Esquema del funcionamiento.

Capítulo 4

Planificación del proyecto

Índice del capítulo

4.1. Metodología

La metodología que se ha seguido en el desarrollo del proyecto es Kanban. Esta es una metodología ágil, muy visual, que nos permite observar las tareas que componen un proyecto y la fase en la que está en cada momento. Kanban se caracteriza por organizarse en un tablero donde se exponen todas las tareas. Hay muchas formas de gestionar el tablero. En este caso particular, se ha dividido en tres como se puede observar en la figura 4.1. Una primera columna con las tareas a realizar, una segunda con una tarea que corresponde a la tarea que se está ejecutando en el momento y una tercera con las tareas ya finalizadas, de forma que sigue un flujo de trabajo de izquierda a derecha.

A lo largo del desarrollo se ha podido apreciar que esta forma de trabajar ha sido muy útil, puesto que esperar a terminar una tarea para comenzar otra evita confusiones y focaliza la faena. De esta forma, no se comenzaba una tarea hasta que la anterior había sido terminada con éxito. La mayoría de las veces tras terminar una tarea, esta era revisada con el supervisor para repasar los problemas que hubiesen surgido y decidir, en función a ello, la siguiente tarea a realizar. Esta es una de las mayores ventajas de las metodologías ágiles.

Para tener un seguimiento de las tareas a realizar en el proyecto, que se ven en el apartado siguiente de planificación se ha utilizado el software Jira de Atlassian [2]. Es la aplicación elegida por la empresa porque además de que se puede gestionar fácilmente, también se pueden vincular las tareas a ramas o *commits* del Bitbucket, ya que son del mismo equipo.

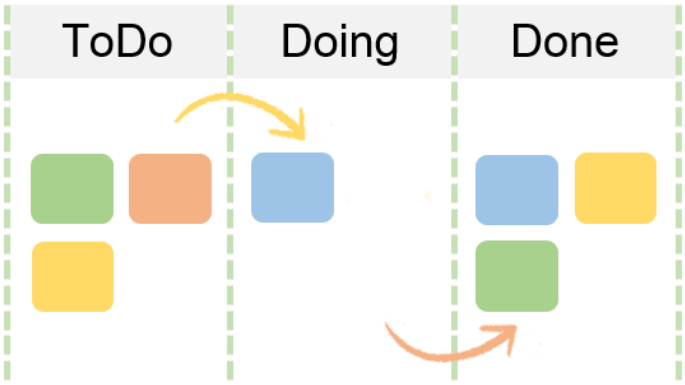

Figura 4.1: Metodología Kanban.

4.1.1. Tareas del proyecto

En la tabla 4.1 se explican detalladamente las tareas a llevar a cabo para la realización del proyecto.

Investigación	
Introducción al proyecto	En esta tarea se realiza la introducción al proyecto junto al tutor del mismo en la empresa, donde se presenta el lugar de trabajo y se facilita toda la información necesaria, cuentas, contraseñas, ordenador, compañeros, etc. Y se repasa el objetivo general del proyecto.
Introducción a Vue	En esta tarea se lee la guía de Vue.js introductoria para aprender los conocimientos básicos del nuevo lenguaje [8].
Introducción a Librería SIP.js	Durante esta tarea se investiga sobre la librería SIP.js puesto que es una herramienta nueva [15].
Introducción a Go	En esta tarea se realiza el tour de Go, consiste en un recorrido interactivo donde se aprende a usar este lenguaje y sus variables [16].
Desarrollo	
Instalación del repositorio de Nayar App	En esta tarea se instala el repositorio de la aplicación en la que se integra la nueva funcionalidad.
Crear página web "Centralita"	Crear la página web en la que se desarrollará el proyecto.
Realizar una llamada	En esta tarea se debe conseguir realizar una llamada desde la página web que sea descolgada por el receptor y se escuche en ambos sentidos.
Recibir una llamada	En esta tarea se debe conseguir recibir una llamada que aparezca en la página web y pueda ser descolgada para mantener una conversación.

Crear interfaz para las llamadas	Esta tarea se requiere para modificar la página web y que sea visualmente atractiva para el usuario, de forma que aparezcan los botones necesarios para llamar, colgar, descolgar, entre otras funciones.
Añadir marcadores DTMF	Esta tarea implica crear botones para cada opción existente en una llamada que requiera pulsar algún número o símbolo.
Crear formulario de la llamada	En esta tarea se ha creado un formulario donde se completa la información relevante a rellenar sobre la llamada.
Enlazar llamada saliente con Odoo	En esta tarea, al marcar un número de teléfono se realizan consultas a Odoo con el fin de obtener información vinculada, como el nombre del cliente y se autocompletan datos del formulario.
Enlazar llamada entrante con Odoo	En esta tarea, cuando hay una llamada entrante, se realiza la consulta a Odoo, para que aparezca el nombre de la persona que llama además del número y se autocompletan datos del formulario.
Registrar datos de la llamada en Odoo	En esta tarea, una vez terminada la llamada y rellenado el formulario, éste se envía a la base de datos de Odoo para almacenar la información.
Incluir gráficos de las llamadas del cliente	En esta tarea, se realiza una consulta de las llamadas del cliente en la última semana y se visualiza para que sirva como referente de la llamada actual.
Incluir gráficos de las llamadas de la empresa	En esta tarea, se realiza una consulta de las llamadas de la empresa en la última semana y se visualiza para que sirva como referente de la llamada actual.

Tabla 4.1: Tareas del proyecto.

4.2. Planificación

Inicialmente el proyecto se planificó siguiendo una propuesta de lo estudiado en una asignatura de la carrera para el desarrollo de proyectos. La planificación se definió para conseguir abordar las tareas designadas por los objetivos de una forma satisfactoria, teniendo en cuenta la limitación de 300 horas para realizarlo.

El proyecto comenzó el 3 de febrero del 2020 y cumpliendo la planificación la fecha de fin estimada era el 5 de mayo del 2020 con el horario acordado entre la empresa y la alumna, de lunes, martes y viernes de 8:00 a 14:00 y miércoles y jueves de 8:00 a 13:00, determinando como festivos el 28 de febrero, el 25 de marzo para la feria de empresas organizada por la UJI y la semana de Pascua, del 13 al 18 de abril.

En la figura 4.2 se pueden observar las tareas que inicialmente se iban a realizar con la fecha de inicio y fin de cada una, además de la conexión de predecesoras de cada una. Más

adelante, en la figura 4.3 se puede ver el diagrama de Gantt donde se aprecia el tiempo de dedicación previsto para cada una de las tareas. Esto resulta muy útil puesto que muestra de forma visual si las tareas se solapan en el tiempo o se realizan secuencialmente, aunque en este caso no se concretaron las horas dedicadas a cada una. Otra característica práctica es que se puede observar la asignación de los recursos humanos a cada tarea, sin embargo este proyecto es exclusivo para una persona, la alumna en prácticas.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Investigación**	**31 días**	**lun 03/02/20**	**lun 16/03/20**	
2	Introducción al proyecto	2 días	lun 03/02/20	mar 04/02/20	
3	Vue.js	2 días	mié 05/02/20	jue 06/02/20	2
4	Librería SIP.js	3 días	vie 07/02/20	mar 11/02/20	3
5	Golang	2 días	vie 14/02/20	lun 17/02/20	4
6	Necesidad de Base de datos	3 días	jue 12/03/20	lun 16/03/20	13
7	**Desarrollo**	**67 días**	**lun 03/02/20**	**mar 05/05/20**	
8	Instalación del repositorio	1 día	lun 03/02/20	lun 03/02/20	
9	Creación de la página "Centralita"	2 días	mar 18/02/20	mié 19/02/20	5
10	Realizar llamada VoIP	5 días	jue 20/02/20	mié 26/02/20	9
11	Recibir llamadas VoIP	5 días	jue 27/02/20	mié 04/03/20	10
12	Crear interfaz para las llamadas	3 días	jue 05/03/20	lun 09/03/20	11
13	Enlazar llamadas al ERP	2 días	mar 10/03/20	mié 11/03/20	12
14	Registro de datos	12 días	mar 17/03/20	mié 01/04/20	6
15	Diseñar base de datos	9 días	mié 08/04/20	lun 20/04/20	14
16	Almacenar llamadas en la base de datos	11 días	mar 21/04/20	mar 05/05/20	15

Figura 4.2: Diagrama de Gantt previo.

La planificación inicial es una estimación porque se desconocen los inconvenientes que pueden suceder durante el proceso de desarrollo de un proyecto. Concretamente este año se ha decretado un estado de alerta inesperadamente tras los sucesos del COVID-19. Por ello, la empresa decidió detener el proyecto durante dos semanas, lo que en un principio se pensaba que duraría (del 16 al 27 de marzo). Como la alarma se prolongó, la empresa propuso retomar el proyecto telemáticamente, de forma que se solicitó una modificación en el horario de trabajo de lunes a viernes de 8:00 a 14:00, además de ser eliminados los festivos de marzo y abril. Esto resultó un cambio en la fecha de finalización al 30 de abril del 2020.

Esta tesitura llevó a un cambio en la planificación prevista para las últimas semanas. Cabe mencionar que se invirtieron algunas horas en realizar el cambio a remoto por temas del servidor utilizado para gestionar las llamadas. Todo el material necesario fue enviado al nuevo lugar de trabajo. Por lo tanto, en la figura 4.4 se observa la planificación de tareas que se realizó finalmente para desarrollar el proyecto. Esta planificación no fue únicamente modificada por la situación, si no que algunos cambios se determinaron a lo largo del proceso, como descartar la creación de una base de datos. Otro de los problemas que retrasó un poco el inicio fue por parte del servidor. Éste era proporcionado y gestionado por la empresa, pero al principio generó un pequeño retraso ya que no conectaba bien con el servidor PBX utilizado y se pensaba que era problema al generar el cliente SIP. Finalmente se encontró el problema, se hizo un cambio a otro servidor que conectó correctamente y se pudo continuar sin una considerable demora. En

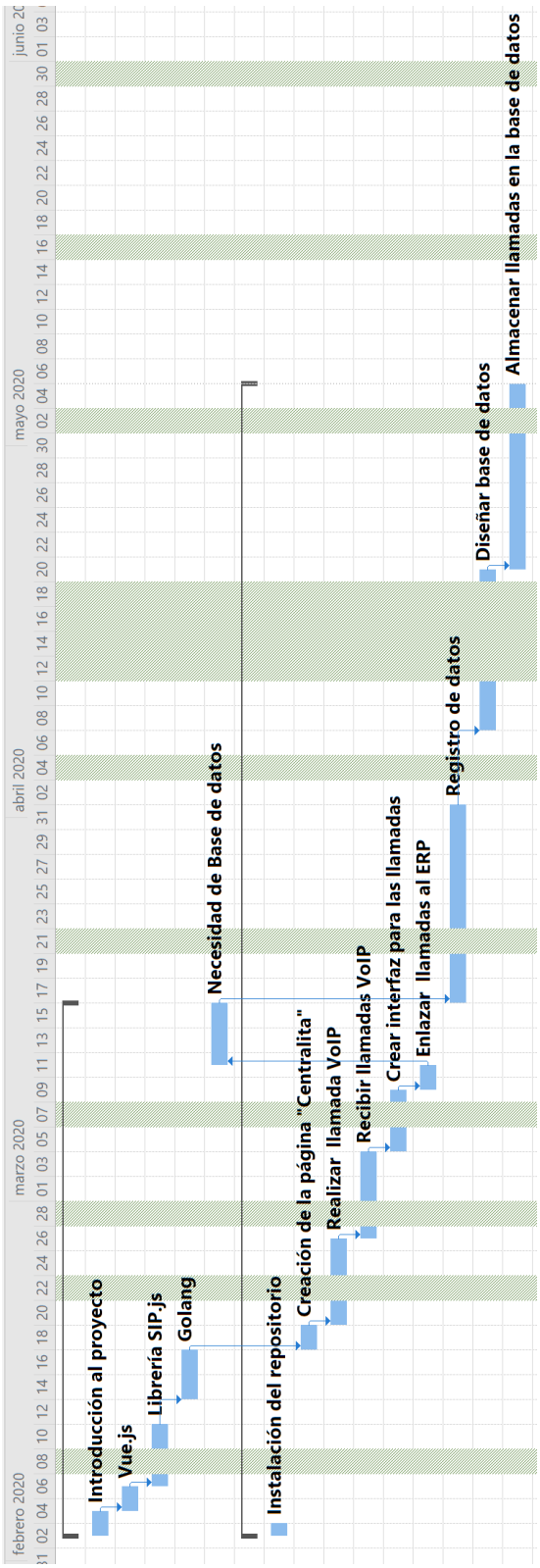

Figura 4.3: Diagrama de Gantt previo.

la figura 4.5 se puede ver el nuevo diagrama de Gantt resultante de los cambios.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Investigación**	**28 días**	**lun 03/02/20**	**mié 11/03/20**	
2	Introducción al proyecto	1 día	lun 03/02/20	lun 03/02/20	
3	Introducción a Vue	3 días	mar 04/02/20	jue 06/02/20	2
4	Introducción a Librerías SIP.js	2 días	vie 07/02/20	lun 10/02/20	3
5	Introducción a Golang	3 días	lun 09/03/20	mié 11/03/20	13
6	**Desarrollo**	**64 días**	**lun 03/02/20**	**jue 30/04/20**	
7	Instalación del repositorio de Nayar App	1 día	lun 03/02/20	lun 03/02/20	
8	Crear página web "Centralita"	2 días	mar 11/02/20	mié 12/02/20	4
9	Realizar una llamada	5 días	jue 13/02/20	mié 19/02/20	8
10	Recibir una llamada	5 días	jue 20/02/20	mié 26/02/20	9
11	Crear interfaz para las llamadas	3 días	jue 27/02/20	lun 02/03/20	10
12	Añadir marcadores DTMF	1 día	mar 03/03/20	mar 03/03/20	11
13	Crear formulario de la llamada	3 días	mié 04/03/20	vie 06/03/20	12
14	Enlazar llamada saliente con Odoo	15 días	jue 12/03/20	mié 01/04/20	5
15	Enlazar llamada entrante con Odoo	2 días	jue 02/04/20	vie 03/04/20	14
16	Registrar datos de la llamada en Odoo	10 días	lun 06/04/20	vie 17/04/20	15
17	Incluir gráficos de las llamadas del cliente	5 días	lun 20/04/20	vie 24/04/20	16
18	Incluir gráficos de las llamadas de la empresa	4 días	lun 27/04/20	jue 30/04/20	17

Figura 4.4: Diagrama de Gantt final.

4.3. Estimación de recursos y costes del proyecto

En cuanto a los recursos necesarios para el desarrollo de un proyecto de este tipo pueden clasificarse en personas, componentes software reutilizables y herramientas de hardware/software necesarios para realizar el proyecto. En el caso de este proyecto los costes se pueden advertir en:

- **Recursos humanos.** Este proyecto se ha planificado para ser desarrollado únicamente por un estudiante durante su estancia en prácticas de 300 horas. Además, lo realiza sin recibir una compensación económica por lo que los recursos humanos son siempre los mismos a lo largo de todo el desarrollo. Una vez lanzada la aplicación es necesario realizar el mantenimiento de la misma en el caso de que surjan errores, que pueden ser cosas como aplicar cambios de seguridad en el código o servidor, o añadir funcionalidades a la página. No obstante, la estimación de los recursos humanos necesarios para el mantenimiento no supera las 10 h mensuales y será responsabilidad del personal de la propia empresa para la que se ha desarrollado el proyecto.
- **Herramientas de hardware y software.** Debemos diferenciar entre las herramientas necesarias durante la fase de desarrollo y las necesarias cuando la aplicación ya se encuentra en producción.
  - Durante la fase de desarrollo es necesario como hardware un equipo informático para la estudiante donde desarrollar la aplicación, con un coste aproximado de 1.200 €

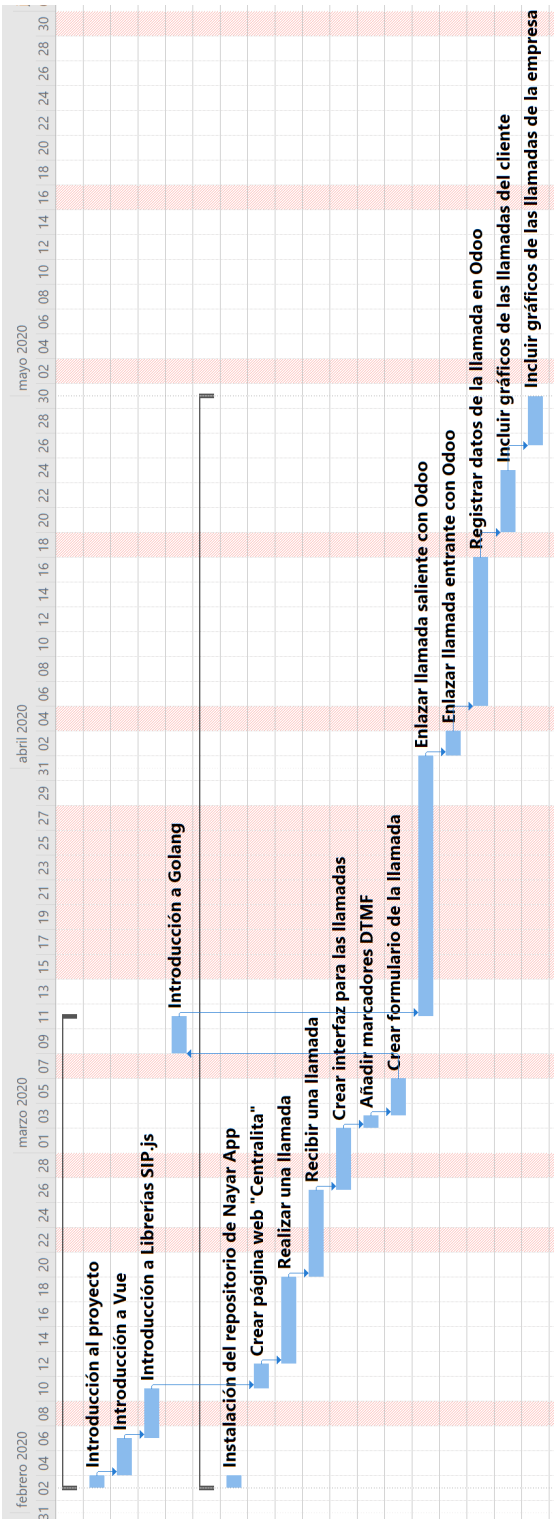

Figura 4.5: Diagrama de Gantt final.

contando con las herramientas imprescindibles. Además se necesita un teléfono con el que realizar las pruebas de las llamadas. La empresa ya contaba con una línea telefónica para otros proyectos, en cambio si proporcionó un teléfono móvil Alcatel Onetouch de 25 €.

- En cuanto al software, es necesario disponer de Visual Studio Code, desarrollador gratuito, por lo que no supone ningún desembolso económico. Para la ejecución de la aplicación durante el desarrollo en el equipo local se debe instalar también el Servidor PBX Issabel, que también es gratuito. También se necesita una cuenta en Odoo, que tiene un coste mensual de 178 €, sin tener en cuenta la implantación ya que esta se realizó anteriormente a este proyecto.

- **Componentes de software reutilizables.** Como se ha mencionado anteriormente, se parte de una aplicación existente donde este proyecto es una funcionalidad añadida.

Cálculo de los costes Aunque el proyecto se realiza por un estudiante sin recibir una compensación económica de la empresa, se puede estimar el coste del desarrollo de la aplicación como si fuera un proyecto profesional teniendo en cuenta las horas invertidas y el precio medio de un programador junior en la zona. Se puede decir que el presupuesto está formado por el coste de desarrollo y el coste de alojamiento y mantenimiento. Sabiendo que la duración de la estancia en prácticas es de 300 horas y cobrando un precio medio de 20 € la hora como programador junior en Castellón, el coste del desarrollo serían 300 x 20 = 6.000 €.

En cuanto al mantenimiento, según lo estimado anteriormente, se deben dedicar 10 horas mensuales a la corrección de errores o mejoras de seguridad, con un coste de 200 €.

Respecto al ERP, el precio por mes se ha determinado que es de 178 €.

4.4. Seguimiento del proyecto

Como se ha explicado en el apartado 4.1, se ha seguido una metodología ágil llamada Kanban. Para hacer un seguimiento de las tareas, la empresa utiliza la ayuda de la aplicación Jira. Esta es una herramienta en línea para administrar las tareas del proyecto, el seguimiento de errores e incidencias, y la gestión de procesos, gracias a sus funciones para la organización de flujos de trabajo. En ella se documentaban todas las tareas, se indicaba cual era la tarea en producción y una vez terminada se marcaba como terminada.

Al mismo tiempo, todos los días, el supervisor en la empresa, Aitor Guerrero, hacia un seguimiento de los progresos conseguidos, además de ofrecer su ayuda cuando convenía. De esta forma, asiduamente se recordaba cuáles eran las tareas realizadas, los problemas a abordar y las siguientes acciones. Además, se realizaron dos reuniones con el que será el "cliente final" para mostrar los avances y este añadía comentarios de posibles mejoras en cuanto a eficiencia, localización de algunos botones, o cómo mostrar la información facilitada. También se comentaban posibles funcionalidades nuevas.

Por otro lado, cada dos semanas, la alumna redactaba informes donde documentaba e informaba a la tutora del proyecto, la profesora María José Aramburu, de los avances conseguidos y

los objetivos a emprender durante las siguientes semanas.

Capítulo 5

Análisis y diseño del sistema

Índice del capítulo

Este capítulo trata sobre el estudio de los requisitos en cuanto a funcionalidades del sistema y el diseño de la implementación de los mismos en sus distintas etapas. Las técnicas de análisis utilizadas son las mismas que se han estudiado en algunas de las asignaturas del grado, como pueden ser diagramas de casos de uso, diagramas de actividades, requisitos de datos y tecnológicos, diagramas de clases y capas de la aplicación y diagramas de navegación para la presentación.

5.1. Análisis del sistema

Diagrama de casos de uso
Un diagrama de casos de uso representa las funcionalidades que debe tener un sistema software asociadas a los actores que se ven afectados por ellas, bien porque deben poder aprovecharse de estas o porque se ven envueltos en alguna fase de su ejecución. Estos actores pueden ser humanos, entidades externas o equipos como servidores u otros productos de software de los que depende la funcionalidad. En la figura 5.1 se muestra el diagrama de casos de uso que corresponde con este proyecto.

Para comprender la figura 5.1, se tiene en cuenta que los dos actores que interactúan con los casos de uso tienen un papel definido y son los siguientes:

- Usuario: Representa al personal de soporte técnico de la empresa que será el que en su momento tendrá acceso a la aplicación web.
- Cliente: Son los clientes de la empresa con los que se intercambian las llamadas.

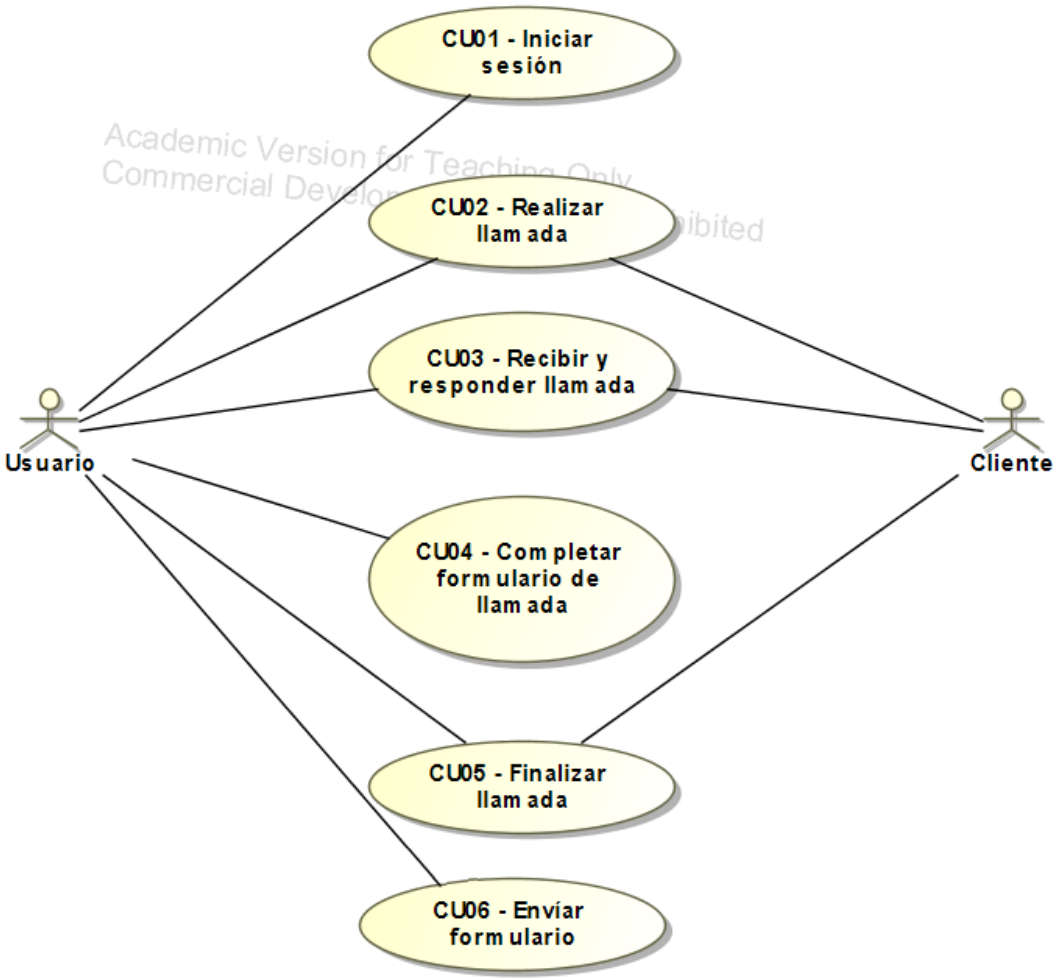

Figura 5.1: Diagrama de casos de uso.

Especificación de casos de uso
La especificación de un caso de uso describe con detalle la funcionalidad que ha de satisfacer el caso de uso, así como las acciones y los pasos detallados que un usuario deberá realizar para utilizar tal funcionalidad. El detalle en la especificación de un caso de uso facilita su posterior implementación y evita posibles malentendidos con el cliente en la fase de especificación de requisitos.

A continuación se presenta la especificación de los casos de uso del proyecto.

Especificación del caso de uso	
ID	CU01
Nombre	Iniciar sesión
Versión	1.0
Autores	María Mañes Velasco
Fecha actualización	10/2/2020
Descripción	El sistema permitirá al usuario inicia la sesión en la plataforma web de la empresa introduciendo sus datos personales manualmente.
Alcance	Desde la introducción de datos en el sistema hasta poder iniciar sesión con éxito.
Actor principal	Usuario
Actores secundarios	-
Condición de final con éxito	El usuario accederá a la aplicación y podrá tener acceso a las funcionalidades asignadas según sus permisos o rol de usuario.
Condición de final con error	El usuario no podrá acceder a la aplicación.
Trigger	Una persona quiere acceder a la aplicación con sus credenciales para utilizar el sistema.
Secuencia normal	**Acción**
1	El usuario introducirá sus claves de acceso a la aplicación e iniciará sesión.
Frecuencia esperada	1 vez al día
Importancia	Necesario
Prioridad	Corto plazo
Comentarios	Sin comentarios adicionales.

Tabla 5.1: Especificación del CU01 - Iniciar sesión.

Especificación del caso de uso	
ID	CU02
Nombre	Realizar llamada
Versión	1.0
Autores	María Mañes Velasco
Fecha actualización	10/2/2020
Descripción	El sistema permitirá al usuario introducir un número de teléfono y llamar.
Alcance	Desde introducir el número de teléfono hasta conseguir la llamada de una manera exitosa, en la que se puede mantener una conversación con el destinatario.
Actor principal	Usuario
Actores secundarios	Cliente
Condición de final con éxito	Establecer conexión telefónica exitosa cuando la llamada es descolgada.
Condición de final con error	No conseguir una conexión telefónica o que ésta no sea descolgada.
Trigger	Alguien desea realizar una llamada.
Secuencia normal	**Acción**
1	El usuario introducirá por escrito el número de teléfono con el que desea contactar.
2	El usuario pulsará un botón que realizará la llamada.
3	El destinatario descolgará y la llamada comenzara.
4	El usuario tendrá la opción mediante un botón de colgar la llamada cuando lo desee.
Frecuencia esperada	Varias veces al día
Importancia	Muy importante.
Prioridad	Corto plazo
Comentarios	Solo los usuarios que tengan credenciales podrán acceder a la aplicación web para realizar las llamadas, pero cualquier cliente puede realizar la llamada desde cualquier otro medio, teléfono, VoIP o cualquier tecnología que realice llamadas.

Tabla 5.2: Especificación del CU02 - Realizar llamada.

Especificación del caso de uso	
ID	CU03
Nombre	Recibir y responder llamada
Versión	1.0
Autores	María Mañes Velasco
Fecha actualización	10/2/2020
Descripción	El sistema permitirá a un usuario registrado descolgar una llamada entrante.
Alcance	Desde recibir una llamada entrante hasta descolgarla para conseguir una conexión que permita la comunicación.
Actor principal	Usuario
Actores secundarios	Cliente
Condición de final con éxito	Establecer conexión telefónica exitosa cuando la llamada es descolgada por el usuario.
Condición de final con error	Que no se muestre la llamada entrante o que no se descuelgue correctamente consiguiendo una comunicación.
Trigger	Alguien desea descolgar una llamada entrante.
Secuencia normal	**Acción**
1	El usuario pulsará un botón que descolgará la llamada entrante.
2	El usuario tendrá la opción mediante un botón de colgar la llamada cuando lo desee.
Frecuencia esperada	Varias veces al día
Importancia	Muy importante
Prioridad	Corto plazo
Comentarios	Solo los usuarios que tengan credenciales podrán acceder a la aplicación web para recibir llamadas, pero cualquier cliente puede recibir las llamadas con normalidad en su dispositivo propio.

Tabla 5.3: Especificación del CU03 - Recibir y responder llamada.

Especificación del caso de uso	
ID	CU04
Nombre	Completar formulario de la llamada
Versión	1.0
Autores	María Mañes Velasco
Fecha actualización	10/2/2020
Descripción	El sistema permitirá al usuario rellenar un formulario que se activa al iniciar la llamada, alguna información se completa automáticamente y otra es introducida por el usuario.
Alcance	Desde que se descuelga una llamada, hasta que el formulario es rellenado.
Actor principal	Usuario
Actores secundarios	-
Condición de final con éxito	Al ser descolgada la llamada aparece el formulario con información completada automáticamente y otra que es rellenada por el usuario. Al mismo tiempo, justo debajo aparece un gráfica con las llamadas de la última semana del cliente y la empresa con la que se mantiene la conversación.
Condición de final con error	Que no aparezca el formulario al descolgar la llamada. Que no aparezca la información completada automáticamente o aparezca mal. Que no permita introducir o seleccionar la información al usuario.
Trigger	Llamada descolgada.
Secuencia normal	**Acción**
1	Descolgar llamada.
1.1	La llamada es saliente y es descolgada por el cliente.
1.2	La llamada es entrante y descolgada por el usuario.
2	Se completa la información que es posible completar.
Frecuencia esperada	Tantas como llamadas
Importancia	Muy importante
Prioridad	Corto plazo
Comentarios	Los datos que se completan automáticamente son la información que ya se conoce y está almacenada en Odoo. Son algunos como, el nombre del cliente, de la empresa u otros datos como la duración de la llamada. Y los datos que debe introducir el usuario son la categoría, el tipo y un apartado donde poder escribir lo que crea necesario. Las gráficas que aparecen son datos almacenados en la base de datos, sobre la categoría de las llamadas de la última semana.

Tabla 5.4: Especificación del CU04 - Completar formulario de la llamada.

Especificación del caso de uso	
ID	CU05
Nombre	Finalizar llamada
Versión	1.0
Autores	María Mañes Velasco
Fecha actualización	10/2/2020
Descripción	La llamada es finalizada porque uno de los dos ha colgado.
Alcance	El cliente o el usuario pulsa el botón de colgar y se finaliza la llamada.
Actor principal	Usuario
Actores secundarios	Cliente
Condición de final con éxito	La llamada finaliza correctamente para ambos.
Condición de final con error	La llamada no finaliza para ninguno de los implicados. La llamada no finaliza para uno de los dos extremos.
Trigger	Pulsar el botón de colgar.
Secuencia normal	**Acción**
1	Pulsar botón de colgar.
1.1	El usuario pulsa un botón en la página web habilitado para finalizar la llamada.
1.2	El cliente pulsa el botón que tenga predeterminado para colgar desde el dispositivo que interacciona con la llamada.
2	La llamada de finaliza.
Frecuencia esperada	Tantas como llamadas
Importancia	Muy importante
Prioridad	Corto plazo
Comentarios	Sin comentarios adicionales.

Tabla 5.5: Especificación del CU05 - Finalizar llamada.

Especificación del caso de uso	
ID	CU06
Nombre	Enviar formulario
Versión	1.0
Autores	María Mañes Velasco
Fecha actualización	10/2/2020
Descripción	Enviar formulario rellenado a la base de datos.
Alcance	Desde enviar el formulario hasta que se almacene en la base de datos como corresponda.
Actor principal	Usuario
Actores secundarios	-
Condición de final con éxito	La información del formulario se ha almacenado correctamente en la base de datos.
Condición de final con error	Los datos del formulario no se almacenan correctamente.
Trigger	Pulsar botón enviar.
Secuencia normal	**Acción**
1	Pulsar botón enviar.
2	Los datos se envían a la base de datos y se transforman para almacenarlos como corresponda.
3	Los datos han sido almacenados y pueden ser accedidos en cualquier momento.
Frecuencia esperada	Tantas como llamadas
Importancia	Muy importante
Prioridad	Corto plazo
Comentarios	El botón de enviar aparece una vez la llamada ha finalizado, ya que los datos no pueden ser almacenados hasta que la llamada no ha terminado, puesto que uno de los datos es la duración.

Tabla 5.6: Especificación del CU06 - Enviar formulario.

Flujo de datos
Para comprender mejor la interrelación entre los casos de uso y el orden en que se usan, se muestra a continuación la figura 5.2. Esta figura indica los pasos a seguir por el usuario en la aplicación web, desde el punto de vista de los casos de uso mencionados anteriormente.

Diagrama de clases El diagrama de clases es un tipo de diagrama de estructura estática en Lenguaje Unificado de Modelado (UML) que describe la estructura de los atributos que las forman y las relaciones entre los objetos. Es un diagrama vital en el apartado del diseño del sistema, puesto que es la base sobre la que se desarrolla la estructura del programa.

Para la realización de este proyecto la base de datos con las clases ya existía, y solo han hecho falta definir algunos de los atributos, que son los que se pueden observar en la figura 5.3. Estos han sido utilizados ya bien sea para buscar información o para almacenarla como nueva. Los nombres están en inglés puesto que tanto la base de datos como el código se realizó en este idioma.

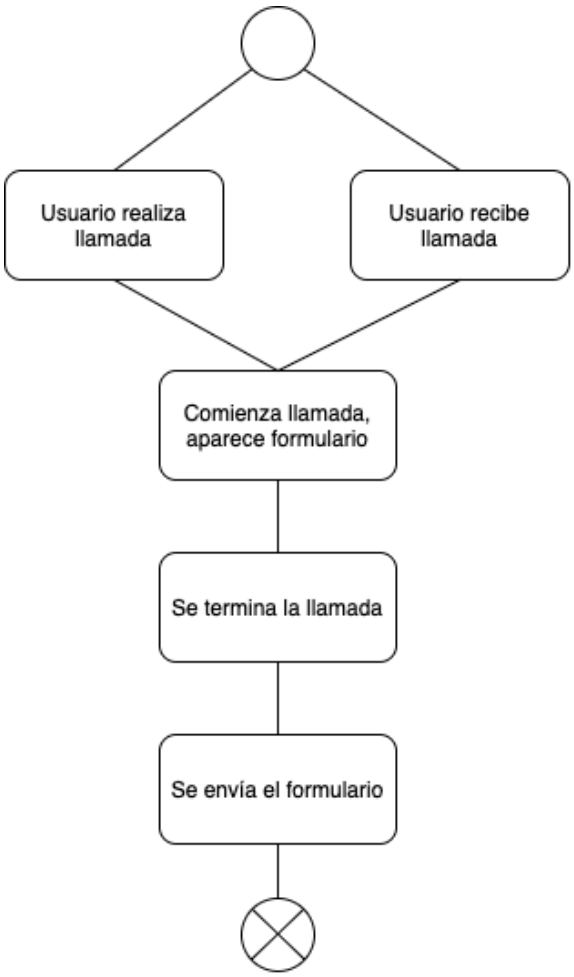

Figura 5.2: Diagrama de flujo de datos.

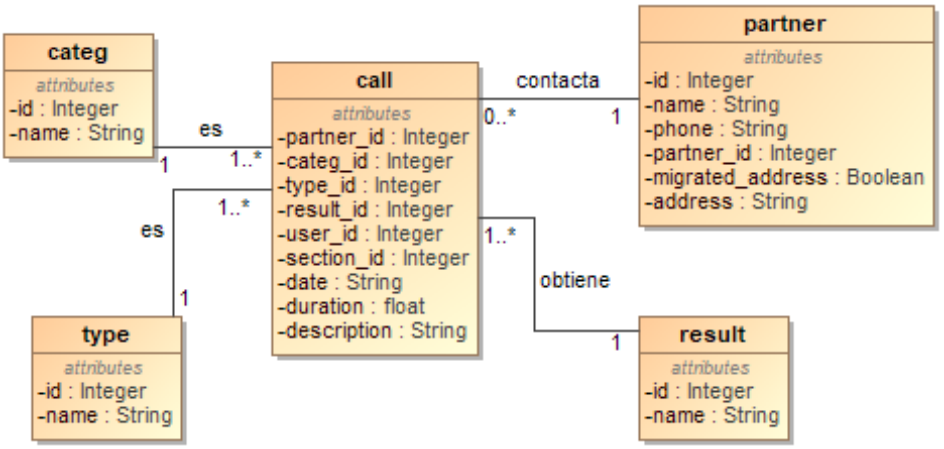

Figura 5.3: Diagrama de clases.

5.2. Diseño de datos

Como bien se ha comentado anteriormente la base de datos ya existía, por lo tanto hubo que realizar únicamente un estudio de su funcionalidad para saber qué datos almacenar y cómo, en qué formato y cuáles son requeridos.

En la figura 5.3 se observan los datos de la base de datos que son necesarios en este proyecto. Para comprenderlo mejor se van a explicar.

5.2.1. Call

La entidad *Call* (llamada) contiene información sobre la llamada, bien sea de entrada o de salida.

Nombre	Tipo	Descripción
partner_id	Integer	Identificador del cliente.
categ_id	Integer	Identificador de la categoría.
type_id	Integer	Identificador del tipo.
result_id	Integer	Identificador del resultado.
user_id	Integer	Identificador del usuario de soporte.
section_id	Integer	Identificador de la sección que gestiona la llamada. En este proyecto se toma por defecto al personal del soporte técnico.
date	String	Fecha y hora de la llamada.
durarion	Float	Duración.
description	String	Descripción o nota escrita por el usuario.

Tabla 5.7: Entidad Llamada.

5.2.2. Partner

La entidad *partner* (cliente) contiene información sobre el cliente y la empresa con la que se comunica mediante la llamada.

Nombre	Tipo	Descripción
id	Integer	Identificador del cliente.
name	String	Nombre del cliente.
phone	String	Número de teléfono del cliente.
partner_id	Integer	Identificador de la empresa a la que representa.
migrated_address	Boolean	Indica si la dirección ha sido migrada (Este dato es necesario por otras funcionalidades y se debe observar su estado).
address	String	Dirección de la empresa.

Tabla 5.8: Entidad Cliente.

5.2.3. Categ

La entidad *Categ* (categoría) contiene información sobre la categoría de la llamada. Ésta puede ser, entrante, saliente, etc.

Nombre	Tipo	Descripción
id	Integer	Identificador de la categoría.
name	String	Nombre de la categoría.

Tabla 5.9: Entidad Categoría.

5.2.4. Type

La entidad *type* (tipo) contiene información sobre el tipo de la llamada. Para identificar esta opción de la llamada la empresa tiene en la base de datos una lista de razones por las que pueden suceder las llamadas y de este modo realizar búsquedas útiles u organizarlas. En el caso de este proyecto, además de almacenarlas, también nos es útil su consulta para realizar las gráficas de las últimas llamadas.

Nombre	Tipo	Descripción
id	Integer	Identificador del tipo.
name	String	Nombre del tipo.

Tabla 5.10: Entidad Tipo.

5.2.5. Result

La entidad *Result* (resultado) contiene información sobre la resolución de la llamada. El resultado de la llamada categoriza cómo ha sido resuelta la llamada según el tipo de problema del que se trate.

Nombre	Tipo	Descripción
id	Integer	Identificador del resultado.
name	String	Nombre del resultado.

Tabla 5.11: Entidad Resultado.

5.3. Diseño de la arquitectura del sistema

Una vez se ha hablado del análisis del sistema, se pasa al diseño de su arquitectura. En la figura 5.4 se puede observar un diagrama con las partes de la aplicación y como fluyen los datos por ella. Se trata de una aplicación web que está formada por una App que conecta el *frontend* y su respectivo *backend* con la parte de los Servicios, donde se encuentra la parte del *backend* que se comunica con la base de datos mediante Nexus.

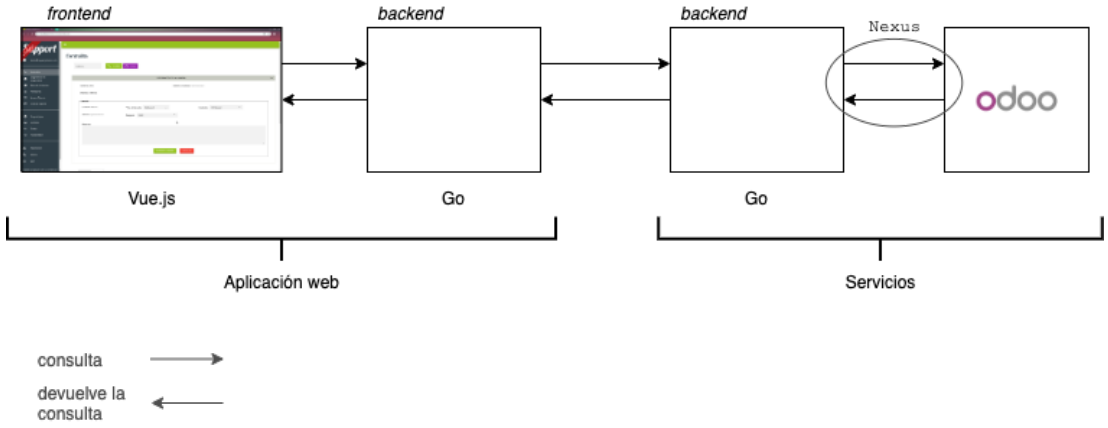

Figura 5.4: Partes de la aplicación.

Lo importante del proyecto está en la aplicación web, en ella la parte del *frontend* que se desarrolla con Vue.js. Este *framework* es progresivo debido a la modularización de las librerías que tiene instaladas. Y la parte del *backend* que se desarrolla en el lenguaje Go. Esta se comunica con los servicios, que también está escrita en lenguaje Go, esta parte es la que conecta mediante Nexus con la base de datos para obtener y guardar la información.

Cabe comentar una serie de aspectos respecto al *frontend*. En primer lugar, los componentes están diseñados bajo el patrón MVVM, explicado anteriormente, lo cual permite que no haya preocupación en cuanto a la renderización del modelo en pantalla.

En segundo lugar, se ha utilizado Quasar [14], este *framework* permite hacer de la página web una aplicación web *responsive* y progresiva. Esto significa que es básicamente una página web, pero mediante el uso de Service Workers y otras tecnologías se comporta más como aplicación normal que como aplicaciones web, ya que puede seguir ejecutándose en segundo plano sin tener que permanecer dentro del navegador.

Por último, se ha utilizado la biblioteca D3.js [5] integrada en Vue.js. Esta biblioteca se utiliza para manipular y visualizar documentos basados en datos utilizando estándares web (HTML, CSS, JavaScript y SVG). D3 permite vincular datos volubles a un Modelo de Objetos de Documento (DOM) del navegador. Esto ofrece una flexibilidad extraordinaria, con una sobrecarga mínima. Además es extremadamente rápido y admite grandes conjuntos de datos y comportamientos dinámicos para la interacción y la animación. La consecuencia de todas estas características es que la manipulación del código es compleja si no se ha tratado antes y requiere una preparación. Por ello, para que su uso entrara en el proyecto, las gráficas son sencillas pero permiten para un futuro una mejora que aproveche esta herramienta.

Respecto al *backend*, en el capítulo 2 ya se ha descrito Nexus, como recordatorio, éste es un servidor utilizado como biblioteca en este proyecto ya que permite la comunicación y sincronización entre múltiples componentes distribuidos, que en éste caso son la aplicación web y Odoo.

5.4. Diseño de la interfaz

Para la realización de la interfaz del usuario, la empresa indicó que como requisito debía seguir los criterios de interfaz de las demás funcionalidades de la aplicación, para cumplir una uniformidad. Por lo tanto, la tipografía y los colores ya estaban definidos.

Se realizaron varios prototipos en papel, principalmente se hizo la figura 5.5 que corresponde al requisito inicial de que la aplicación realizara y recibiera llamadas. Más adelante fueron diseñados los prototipos de las figuras 5.6 y 5.7 para los nuevos requisitos. Estos prototipos sirvieron para conseguir un diseño final, que a pesar de no ser igual, sigue la misma dinámica. Como se puede observar más adelante en las figuras 5.8, 5.9, 5.10, 5.11, 5.12 y 5.13.

Figura 5.5: Prototipo de interfaz 1.

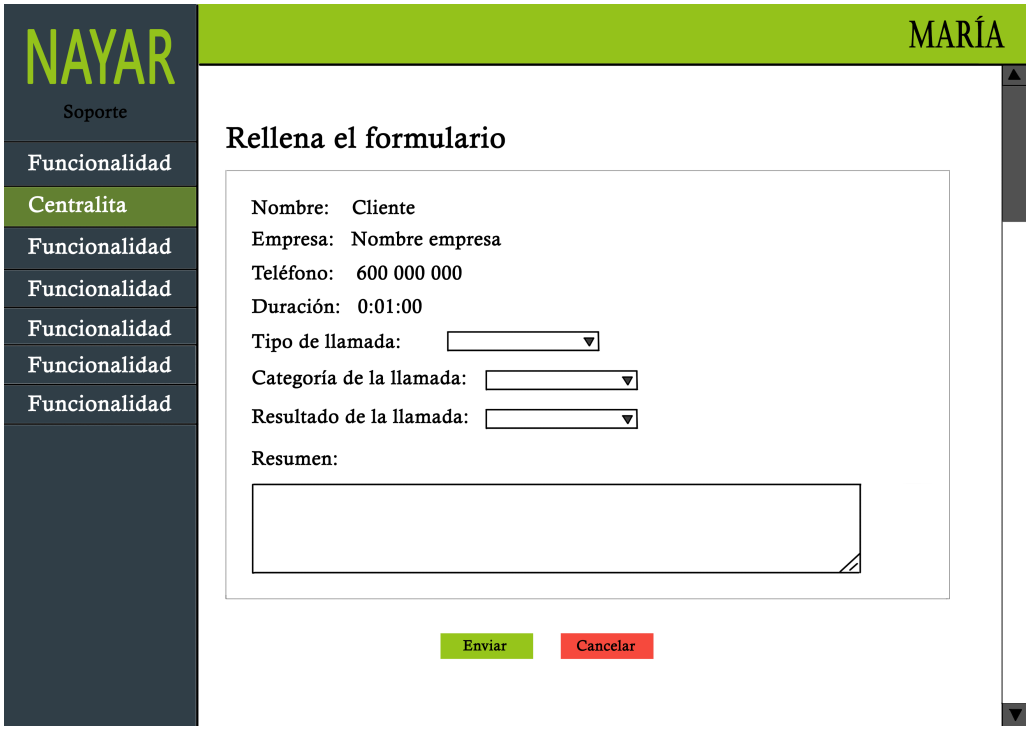

Figura 5.6: Prototipo de interfaz 2.

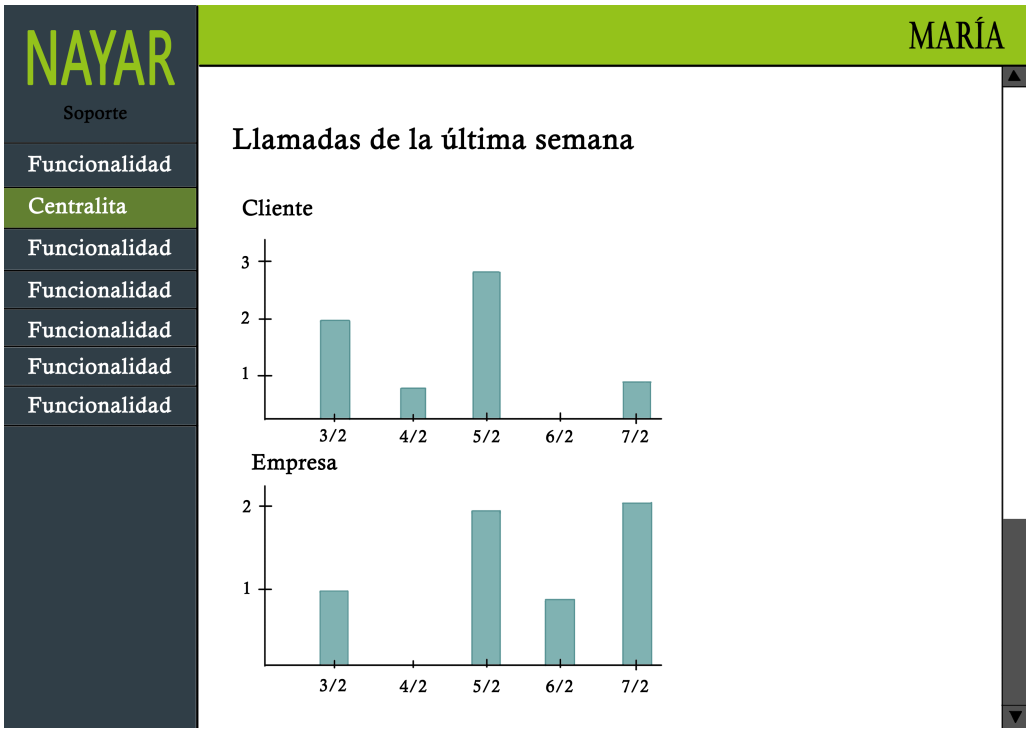

Figura 5.7: Prototipo de interfaz 3.

Tras iniciar sesión en la aplicación, que es una funcionalidad que no entra dentro de este proyecto, se puede acceder a cualquiera de las funcionalidades del menú de la izquierda. En este menú encontramos la funcionalidad de **Centralita** que es la que se ha desarrollado en este proyecto. Como se observa en la figura 5.8 en ella encontramos la opción de introducir el número de teléfono y llamar, o un botón que dirige directamente a la web de Odoo, que como ya se ha nombrado anteriormente, es el ERP de la empresa y a la vez hace de función de base de datos para la información de los clientes, las empresas y sus llamadas.

Si la llamada es entrante aparece un cuadro con el nombre de la persona que está llamando y un botón para descolgar y otro para colgar, esto se puede observar en la figura 5.9. Una vez en llamada, la página cambia a la de la figura 5.10, donde aparece un texto que informa sobre la persona con la que se está comunicando, y aparece además un formulario, que permite la posibilidad de minimizar, para poder observar las gráficas de la figura 5.11 sin tener que deslizarse hasta abajo. En este formulario se habrá rellenado automáticamente información como el nombre del cliente, la empresa y además la duración se puede ver en tiempo real. En el caso de que el cliente no esté registrado aparece la opción de introducir el nombre, como se observa en la figura 5.12. Otros datos que se deben seleccionar a partir de unos desplegables que muestran las posibles opciones según sea la característica, el tipo y la resolución de la llamada, además de existir un apartado donde añadir una descripción escrita.

Como se ha mencionado, bajo este formulario se pueden visualizar dos gráficas, con las llamadas de la última semana del cliente y de la empresa con la que se mantiene la llamada.

Una vez terminada la llamada, el formulario todavía es modificable, pero ya aparece la opción de enviar, de forma que éste se envía para ser almacenado en la base de datos y se vuelve a la página principal lista para realizar o recibir llamadas.

Por último, en la figura 5.13 podemos ver la web a la que dirige el botón de Odoo mencionado anteriormente, éste redirige a la web donde aparece el listado de las llamadas entrantes en Odoo con toda su información.

Figura 5.8: Interfaz de la página principal.

Figura 5.9: Interfaz de llamada entrante.

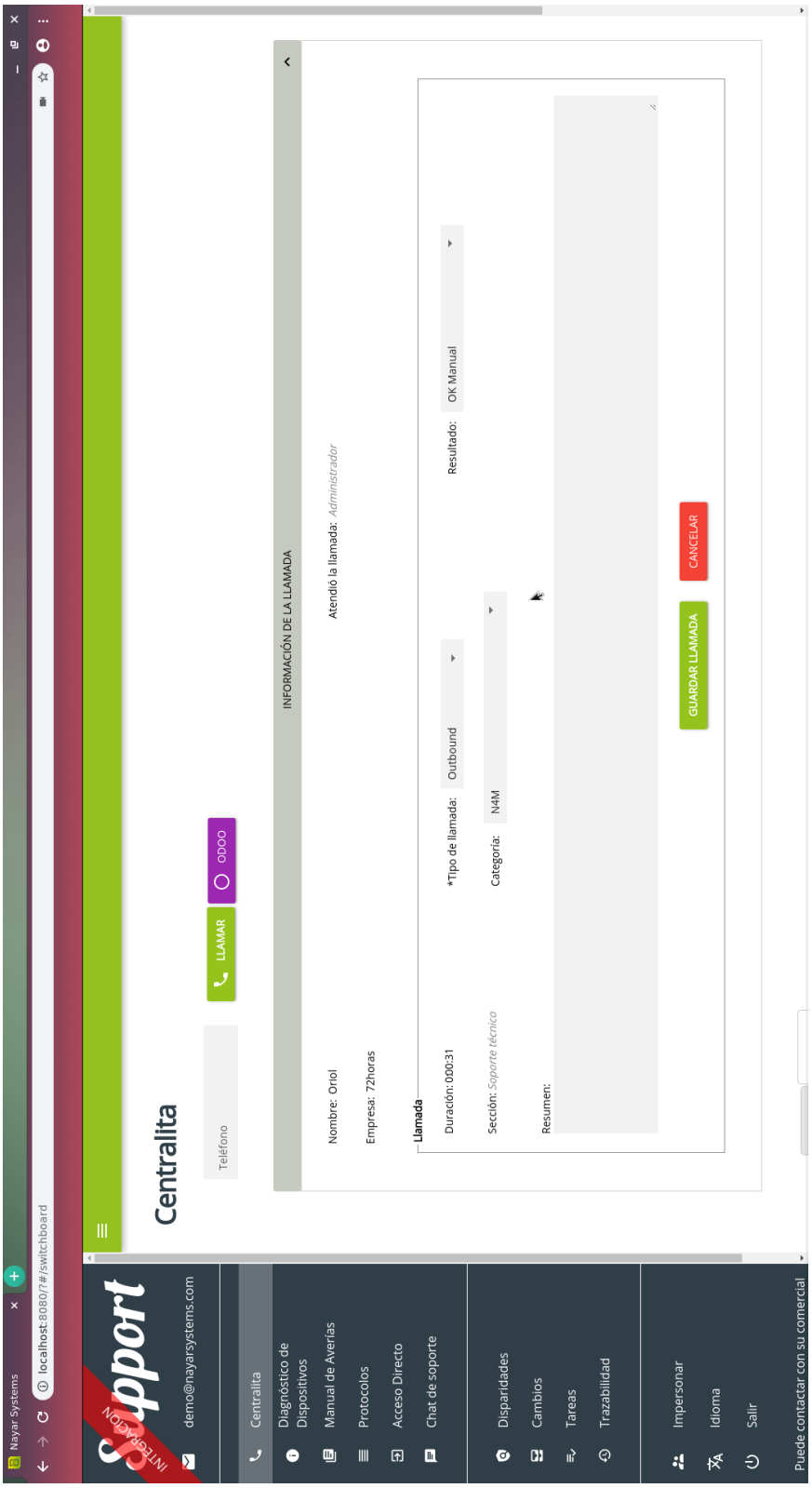

Figura 5.10: Interfaz del formulario tras finalizar la llamada con un cliente ya registrado.

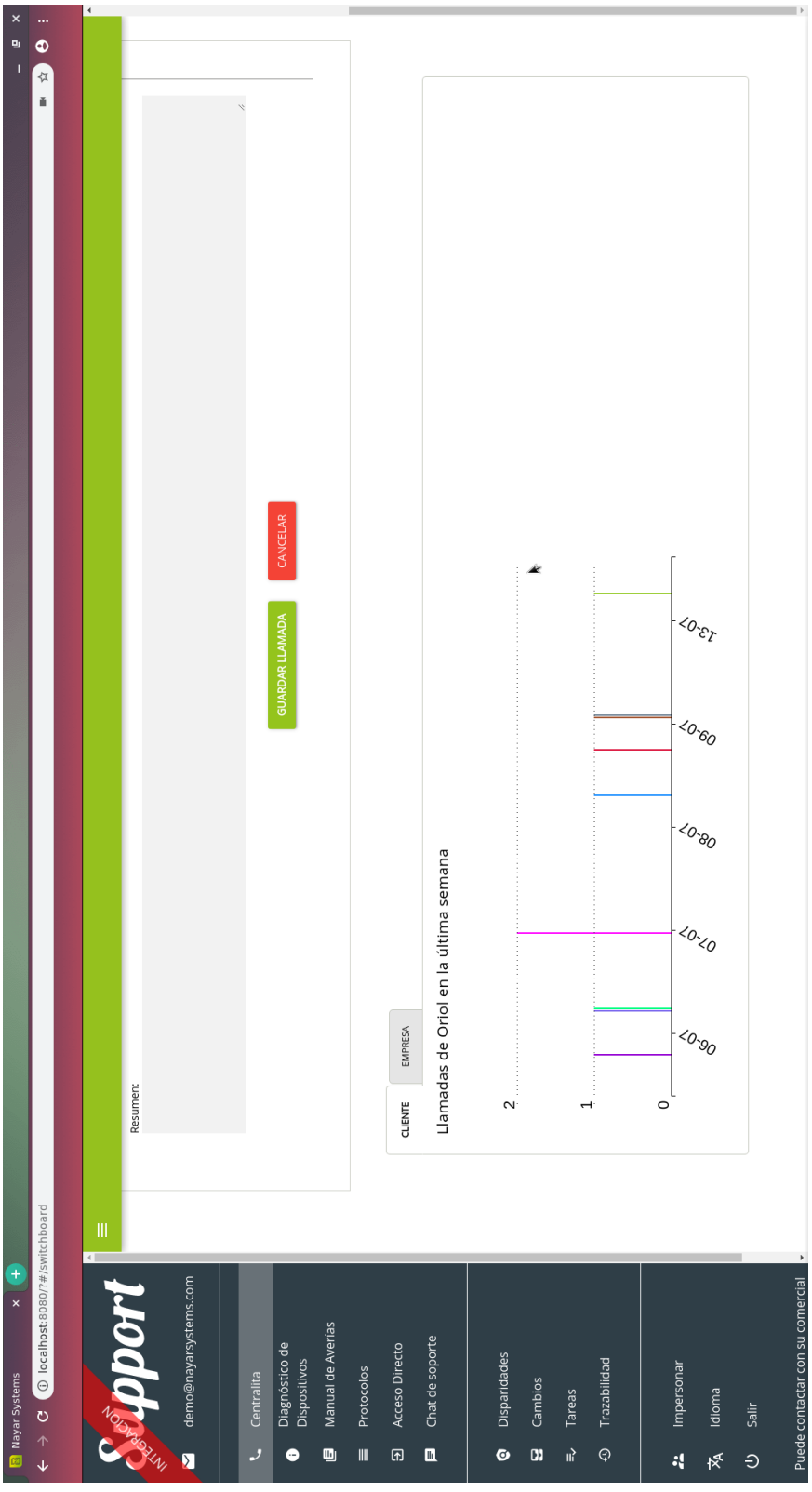

Figura 5.11: Interfaz de la gráfica de las llamadas del cliente la última semana.

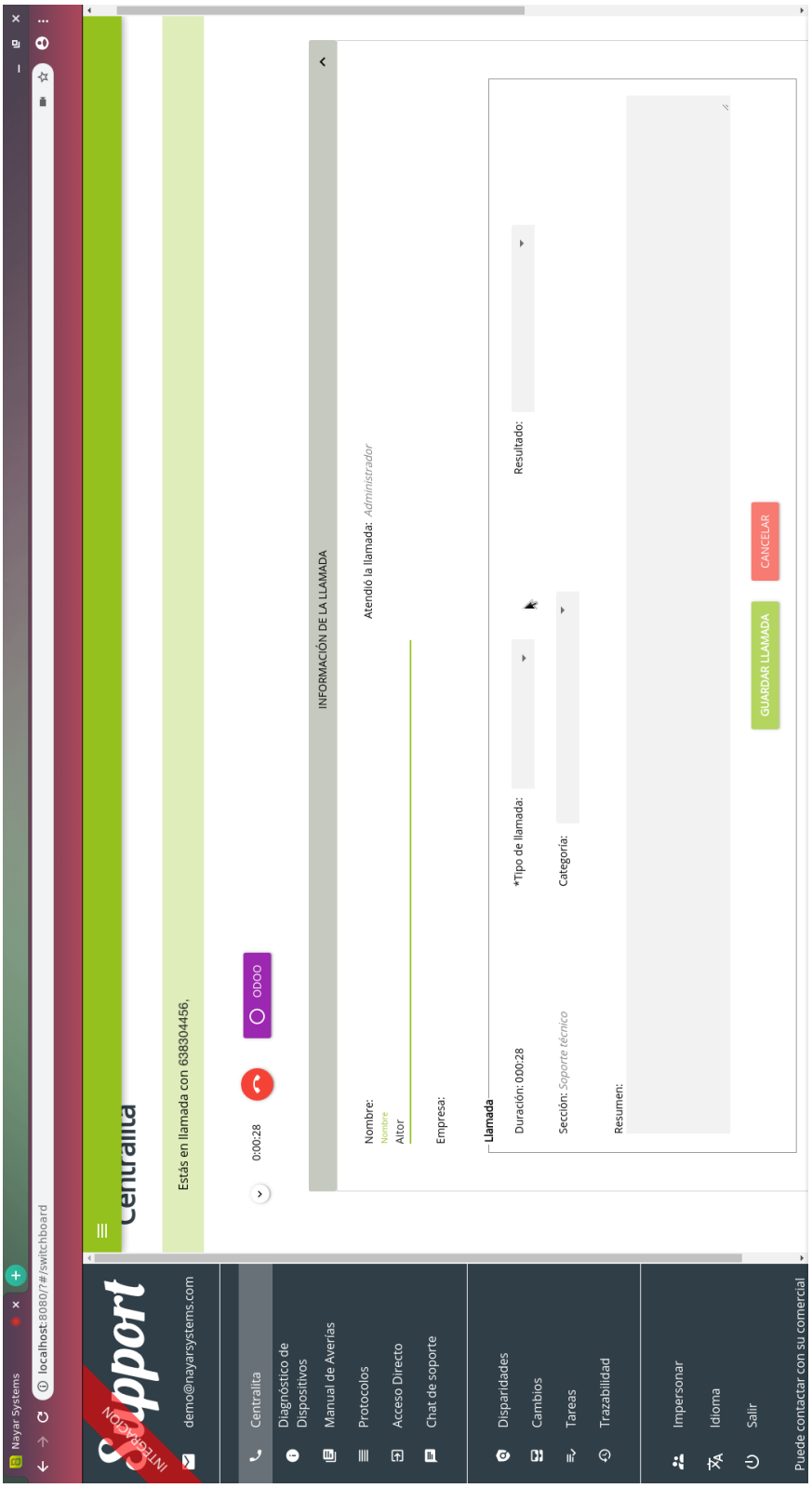

Figura 5.12: Interfaz del formulario durante una llamada de un cliente no registrado.

Figura 5.13: Acceso a la base de datos de las llamadas en Odoo.

Capítulo 6

Implementación y pruebas

Índice del capítulo

En este capítulo se trata en detalle la implementación del proyecto, así como la estructura del código y los componentes de la aplicación y los lenguajes de programación utilizados.

6.1. Detalles de implementación

Como se ha explicado en el subcapítulo 5.3, el lenguaje que más se ha utilizado para el desarrollo de este proyecto es Vue.js, este es un *framework* de JavaScript cada vez más utilizado el mundo del desarrollo *frontend*.

Vue.js proporciona una mejor organización, porque en el mismo componente, puedes tener la estructura HTML correspondiente, la función JavaScript para mejorar su comportamiento y el estilo que contendrá la estructura. Esto hará que la aplicación y la carga de procesamiento sean más rápidas.

Al mismo tiempo, se han utilizado librerías que nos permiten mejorar la aplicación:

- Nexus: Permite la comunicación de microservicios ubicados en distintas localizaciones.
- SIP.js: Esta biblioteca ayuda a introducir el protocolo SIP en la aplicación WebRTC, permitiendo realizar llamadas desde la aplicación web.
- D3.js: Esta biblioteca se utiliza para manipular y visualizar documentos basados en datos utilizando estándares web.

A continuación, en la figura 6.1 se puede observar el formato que se ha seguido para desarrollar el código.

```
<template>
    <default-layout platforms>
        <h1 class="text-bold">{{ $t("SWITCHBOARD_TEL") }}</h1>
        ...
    </default-layout>
</template>
<script>
    import * as SIP from "sip.js";
    ...
    export default {
        components: {
            DefaultLayout: () => import("../../layouts/default.vue")
        },
        data() {
            return {
                /* Inicializar variables necesarias. */
                ...
            }
        },
        created: function() {
            /* Metodos que se activan al cargar la pagina. */
            ...
        },
        methods: {
            /* Metodos que son llamados como consecuencia de una accion. */
            ...
        }
    }
</script>
<style>
    /* Todos los estilos de la pagina. */
</style>
```

Figura 6.1: Estructura del código.

El código se divide en tres bloques. El primero de "template" es en lenguaje HTML, donde se desarrolla toda la parte visual del código con sus correspondientes etiquetas. El ejemplo que se puede observar en el encabezado, en este caso *$t("SWITCHBOARD_TEL")* está escrito en este formato porque accede al dato que muestra según el idioma en el que esté la página.

A continuación está el bloque de "script" donde se encuentran las funciones de la aplicación. Están escritas en lenguaje de Vue que se basa en JavaScript, la diferencia de Vue es que utiliza una sintaxis especial, para comunicarse mejor con el DOM de la web y cargar únicamente la información necesaria. Inicialmente se cargan los datos importados, se puede ver como ejemplo como se importa la librería SIP.js utilizada, pero se han importado otras como Quasar y D3.

A continuación en el apartado "data" se inicializan todas las variables que se van a necesitar más adelante. Seguidamente, el apartado de "created" en el cual están las funciones que se deben ejecutar nada más cargar la web. Se puede acceder a los datos interactivos, aunque no se puede manipular el DOM porque todavía no ha sido montado. Principalmente, se ha creado el agente SIP necesario para realizar y atender las llamadas. Por último, un apartado "methods" que recoge todos los métodos que se activan a partir de que ocurre una acción. En éste mismo apartado se encuentran las consultas al *backend* de la aplicación web cuando se recibe la respuesta se almacena en las variables y se muestra como sea conveniente.

Para terminar, el bloque "script" recoge los estilos CSS que se utilizan para formatear el diseño de la página web.

Por otro lado, para la parte del *backend* se ha utilizado el lenguaje de Go. Tanto en la parte de la aplicación como en la de los servicios, donde se realiza la comunicación mediante Nexus a la base de datos. Este lenguaje es relativamente nuevo, sencillo y eficiente, además existe mucha documentación publicada, lo que ayuda a un novato en este lenguaje a su comprensión y desarrollo.

Como se ha mencionado, el *frontend* genera una consulta al *backend* de la misma aplicación y este lo transmite a los servicios. Éstos reciben la consulta y mediante Nexus generan la misma directamente a la base de datos, que devolverá la información que Go transforma en información útil y en un formato cómodo para que el *frontend* la gestione. Este flujo de datos se ha podido observar de forma gráfica en la figura 5.4.

6.2. Pruebas

Por el tipo de proyecto del que se trata no se ha podido realizar un código de batería de pruebas, si no que éstas se hacían manualmente. Estas pruebas deben comprobar que se cumplen todos los requisitos establecidos.

Por ello, inicialmente se ha comprobado que las llamadas funcionan tanto de entrada como de salida correctamente creando el agente SIP oportuno, y que los datos tanto de entrada como de salida son transmitidos por la red en ambas direcciones. Estas pruebas fueron muy necesarias al inicio del proyecto, ya que como se ha mencionado en la planificación, inicialmente surgieron problemas y no se oía en ambas direcciones. Esto resultó ser un problema del servidor y de que se capturaban los datos antes de que el agente SIP se creara, por lo tanto se perdían.

Otra prueba que se tuvo que tener en cuenta era la comunicación con la base de datos. Al empezar la llamada se realiza una búsqueda de datos a través del número de teléfono al que se llama o el que realiza la llamada. Si este número está registrado en la base de datos se devuelven algunos datos útiles sobre el cliente, por lo tanto se debe comprobar que tanto la búsqueda como la respuesta de estos son correctos y en el caso de que no lo sean se muestre el mensaje pertinente.

A continuación, el siguiente requisito fue realizar un formulario para almacenar la llamada. Para este formulario se utiliza la información obtenida en la primera búsqueda del número de teléfono, y además se hace otra consulta que devuelve unas listas de opciones que se mostraran como selectores donde el usuario elige la más conveniente para la llamada. Tras colgar la llamada se debe rellenar toda la información relevante y esta será enviada a la base de datos. Por lo tanto, se debe comprobar que inicialmente la información que se muestra al usuario es completa y correcta, y para verificar que los datos introducidos y enviados a la base de datos son también correctamente almacenados.

Por último, para las gráficas se debe hacer una consulta de las llamadas tanto del cliente como de la empresa aplicando un filtro que devuelva únicamente las de la última semana, de

forma que no se sobrecargue con información inútil en este caso. Esto conllevó a problemas por la tecnología utilizada D3.js, ya que es compleja y hay que pasarle los datos de una manera muy concreta, por ello había que estar muy pendiente de que los datos que mostraba eran reales.

Capítulo 7

Conclusiones

Índice del capítulo

En este capítulo se exponen los resultados del desarrollo del proyecto y las conclusiones finales.

7.1. Resultados del proyecto

El resultado del proyecto ha sido muy positivo. Se han cumplido todos los objetivos del mismo, incluso se han superado los requisitos iniciales llegando a almacenar las llamadas tras rellenar un formulario y mostrar información relevante.

Durante la estancia en la empresa donde se han desarrollado las prácticas no se ha experimentado ningún problema destacable y la colaboración del supervisor fue constante y eficaz.

En cuanto a la puesta en marcha de la aplicación, todavía necesita ciertas pruebas que no se pudieron realizar puesto que el final del proyecto se realizó desde casa y algunas cosas estaban hechas para que funcionen a través de la VPN de la empresa. Además, la empresa debe realizar algunos cambios en el servidor para integrar el proyecto con otras aplicaciones antes de poder ofrecerla a sus usuarios.

7.2. Conclusiones técnicas

Una vez finalizado el proyecto, haber utilizado SIP.js me ha parecido una decisión acertada. Todos los problemas que han surgido se han solventado con éxito. A pesar de no haber podido elegir los lenguajes de desarrollo como han sido Vue y Go, se ha comprobado que son muy validos para este trabajo puesto que han dado unos buenos resultados tanto en *frontend* como en *backend*. También utilizar Quasar como *framework responsive* ha facilitado mucho el diseño de las vistas para dispositivos pequeños. Al igual que la librería D3.js se ha comprobado que es una gran herramienta para la gestión de datos a la hora de mostrarlos.

Cabe tener en cuenta que la extensión del proyecto no es suficientemente grande como para que las tecnologías empleadas pudieran presentar limitaciones o deficiencias. Aún así, es importante decir que no se ha experimentado ningún problema con respecto a los lenguajes ni herramientas utilizadas.

7.3. Posibles mejoras

Durante el proyecto, siempre surgían posibles funcionalidades que eran muy interesantes para añadir, todas ellas eran muy reales si no fuera por la limitación de las 300 horas. Pero esas mejoras quedaron anotadas para tenerlas en cuenta en un futuro.

Una de las mejoras que se comentó fue añadir un apartado donde aparezcan los usuarios y su estado, para que a la hora de tener que derivar una llamada se viera a simple vista si el otro usuario estaba disponible, en otra llamada o no disponible. También la opción de derivar llamadas, que tiene más cavidad en el servidor y en el PBX, un tema que se comentó como de posible implementación.

Por otra parte se habló de añadir un listado con llamadas no atendidas, ya bien porque el grupo de soporte no estuviera disponible por horario o porque estaban todos atendiendo otras llamadas. En este caso a partir del listado se podría devolver la llamada al cliente.

Se valoró la opción de añadir un apartado con *Helpdesk*, que es el término que le dan a todas las incidencias que reciben. Esta opción se tuvo que descartar por la forma en la que las incidencias se almacenan, que no permite una búsqueda de únicamente incidencias en llamadas.

Finalmente, se consideró la idea de dar de alta a los clientes en la base de datos, pero no se determinó si desde esta misma página o en una nueva funcionalidad.

Estas son las ideas que fueron surgiendo para implementar en el caso de que sobraran horas, además de las que sí se llegaron a realizar, pero esto no quiere decir que no hayan muchas otras posibles mejoras a añadir.

7.4. Conclusiones personales

Llegado este momento, dado que personalmente no había hecho ningún proyecto desde cero con anterioridad, esto suponía afrontar un nuevo reto. El miedo desapareció la primera semana en la empresa, al ver que tanto mi supervisor como todos los compañeros eran muy amables y estaban siempre dispuestos a ayudarme con cualquier duda, lo cual hizo que se amenizara mi estancia y se me quitaran algunas incertidumbres.

Cuando llevaba mes y medio de proyecto comenzó el estado de alerta por el COVID-19, lo que provocó que se paralizara el desarrollo dos semanas, solución con la que estoy de acuerdo porque al retomarlas telemáticamente los demás compañeros de la empresa ya habían estado trabajando, de forma que las cosas estaban más calmadas y podían orientarme mejor. Realizar las prácticas telemáticamente supuso un reto tanto para la UJI, como para la empresa y para mí misma. A consecuencia de ello, las tareas se retrasaron un poco y surgieron algunos problemas los primeros días para conseguir que todo funcionara correctamente con la VPN, etc. Pero estoy contenta porque ahora pienso que estoy más preparada para las adversidades inesperadas de lo que estaba. Y pude comprobar que tenía mucho apoyo tanto por parte de los compañeros como de la tutora.

Respecto al proyecto estoy muy contenta de su resultado, de haber podido hacer más funcionalidades a parte de los requisitos iniciales y me he quedado con ganas de añadirle muchas mejoras, puesto que el proyecto me ha motivado en todo momento y siento que he aprendido mucho sobre tecnologías que no conocía al inicio.

Por último, personalmente estoy muy orgullosa de haber llegado tan lejos y poder decir que casi soy Ingeniera en Informática, ya que muchas veces en el camino he dudado de estar segura con lo que hacía. Sé que al terminar muchos sufrimos el injustificable "síndrome del impostor", pero confío que pronto desaparecerá. Al menos al realizar tanto este proyecto como algunos otros durante la carrera he disfrutado con lo que hacía, y he podido ver qué es lo que más me llena. Por eso quiero agradecer a los profesores que me han formado, a los compañeros con los que he sufrido las horas de estudio y trabajos interminables siempre acompañados de risas, pero sobre todo a mi familia y a mi pareja porque me han apoyado desde el primer momento, no dejándome caer nunca y apostando siempre por mí.

Bibliografía

[1] https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html. [Consulta: 1 de Abril de 2020].

[2] Aplicación de atlassian. https://www.atlassian.com/es/software/jira. [Consulta: 23 de Abril de 2020].

[3] Bitbucket. https://www.atlassian.com/es/software/bitbucket. [Consulta: 19 de Julio de 2020].

[4] Contorl de versiones git. https://git-scm.com/. [Consulta: 19 de Julio de 2020].

[5] D3.js. https://d3js.org/. [Consulta: 1 de Julio de 2020].

[6] ERP Odoo. https://www.odoo.com/es_ES/. [Consulta: 19 de Julio de 2020].

[7] Framewokr Vue.js. https://vuejs.org/. [Consulta: 19 de Julio de 2020].

[8] Guia introductoria Vue. https://es-vuejs.github.io/vuejs.org/v2/guide/. [Consulta: 16 de Junio de 2020].

[9] Lenguaje Go. https://golang.org/. [Consulta: 19 de Julio de 2020].

[10] Lenguaje JavaScript. https://developer.mozilla.org/es/docs/Web/JavaScript. [Consulta: 19 de Julio de 2020].

[11] Machine to machine. https://es.wikipedia.org/wiki/M2M. [Consulta: 1 de Abril de 2020].

[12] Patron Modelo Vista Vista Modelo. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel. [Consulta: 14 de Junio de 2020].

[13] Patron Modelo Vista Vista Modelo según Vue.js. https://012.vuejs.org/guide/#Introduction. [Consulta: 14 de Junio de 2020].

[14] Quasar. https://quasar.dev/. [Consulta: 1 de Julio de 2020].

[15] SIP.js. https://sipjs.com/. [Consulta: 19 de Julio de 2020].

[16] Tour de Go. https://tour.golang.org/welcome/1. [Consulta: 16 de Junio de 2020].

[17] Visual Studio Code. https://code.visualstudio.com/. [Consulta: 19 de Julio de 2020].

[18] José Luis Aracil Gómez del Campo. Nexus. https://github.com/jaracil/nexus/wiki. [Consulta: 23 de Abril de 2020].